REPORT DOCUMENTATION PAGE	*Form Approved* *OMB NO. 0704-0188*

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE Nov 98	3. REPORT TYPE AND DATES COVERED Final 1 July 1996 - 30 June 1998

4. TITLE AND SUBTITLE Intelligent Agents for the Digital Battlefield	5. FUNDING NUMBERS DAAH04-96-1-0297
6. AUTHOR(S) V.S. Subrahmanian and J. Hendler	
7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(ES) University of Maryland College Park, MD 20742	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) U.S. Army Research Office P.O. Box 12211 Research Triangle Park, NC 27709-2211	10. SPONSORING / MONITORING AGENCY REPORT NUMBER ARO 36161.1-MA

11. SUPPLEMENTARY NOTES
The views, opinions and/or findings contained in this report are those of the author(s) and should not be construed as an official Department of the Army position, policy or decision, unless so designated by other documentation.

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution unlimited.	12 b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

In this report we describe some of the theoretical and practical underpinnings of a joint University of Maryland/Army Research Laboratory Project to develop a scalable architecture for supporting intelligent-agent applications. The main focus of our long term research is threefold:

- to develop the theoretical foundations of intelligent agent systems,
- to concurrently build prototype implementations of such agent based systems, and
- to concurrently develop applications in the area of situation awareness and assessment using the proposed agent architecture and implementation.

Continued On Reverse Side

14. SUBJECT TERMS			15. NUMBER IF PAGES
			16. PRICE CODE
17. SECURITY CLASSIFICATION OR REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT

A specific outcome of our long term research will be the development of a *collaborative agent technology system* , CATS, that will provide the underlying software infrastructure needed to build large, heterogeneous, distributed agent applications. CATS will provide a software environment through which multiple intelligent agents may interact with other agents – both human and computational. In addition, CATS will contain a number of intelligent agent components that will be useful for a wide variety of applications.

Final Report
Contract Number DAAH04-96-0297
Army Research Office

Principal Investigators: V. S. Subrahmanian & J. Hendler
Institute for Advanced Computer Studies
University of Maryland
College Park, MD 20742-3275
Phone: 301-405-6722 Fax: 301-314-9658
Email: [vs, hendler]@cs.umd.edu

1 Technical Summary

Abstract

In this report we describe some of the theoretical and practical underpinnings of a joint University of Maryland/Army Research Laboratory Project to develop a scalable architecture for supporting intelligent-agent applications. The main focus of our long term research is threefold:

- to develop the theoretical foundations of intelligent agent systems,
- to concurrently build prototype implementations of such agent based systems, and
- to concurrently develop applications in the area of situation awareness and assessment using the proposed agent architecture and implementation.

A specific outcome of our long term research will be the development of a *collaborative agent technology system* , **CATS**, that will provide the underlying software infrastructure needed to build large, heterogeneous, distributed agent applications. **CATS** will provide a software environment through which multiple intelligent agents may interact with other agents – both human and computational. In addition, **CATS** will contain a number of intelligent agent components that will be useful for a wide variety of applications.

Introduction

The term *Intelligent Agent* has been used extensively in recent years to describe a variety of software programs that exhibit certain (unfortunately usually ill-defined) characteristics. Examples of programs that are called agents in the literature include *news agents* that scan one or more sources of news data and extract articles of interest to a user (based on some profile of the user's interests), *stock agents* that monitor fluctuations in certain stocks designated by a user and alert him when certain trends (defined by the user) are observed, *agents* that monitor movie and theatre listings with a view to sending regular updates of interesting events to the user, as well as agents that can send email to other humans or agents and schedule meetings and the like. As another example, in the very near future, individuals will probably be able to configure agents that "get" their bills (phones, utilities, water, newpaper, credit cards) by accessing appropriate authorized agents (of the phone, water, etc. company), and automatically instructing the bank to transfer money to these agents if certain exception conditions do not occur. Different users may program different conditions (e.g John may say "pay all bills under 25 dollars and email me that they've been paid, but if the bills are over 25 dollars, send the bill to me and pay it only after my approval"; conversely, Lisa may use a very different condition and may want to examine all bills that exhibit a fluctuation of over 20% vis a vis her previous bill).

In the US military, agent technology may be used in situation assessment for monitoring, say, the weather over Bosnia. A pilot may rely on a weather agent as well as a situation agent to tell him/her about expected weather and enemy deployments in a region. In the case of a pilot, the agents may include detailed information about windspeed, barometric pressure, external temperature and precipitation, and suggested flight paths and fuel levels needed to avoid hostile fire as well as inclement weather patterns. A general using the same agents, on the other hand, may expect detailed analyses of enemy troop movements, and how weather conditions may affect his mission. In both cases, certain changes must be communicated to the user (e.g. the pilot or the general), yet the nature of data communicated will differ. For instance, the general may not be interested in windspeed changes *unless* these changes mean that he and/or the enemy will not be able to deploy certain resources. On the other hand, the pilot *must* be informed rapidly of all windspeed changes above a certain threshold.

One aim of our work is to define what it means for an arbitrary software program P to be considered an intelligent agent. We argue that for a program P to be considered an agent, it must have some properties. It must have *awareness* of its own properties (what can I compute ? how fast can I compute it? how accurate are these computations? which other agents can I communicate with? etc.). Second, it must have the ability to *collaborate* with at least some other agents – in other words, it uses a sufficiently rich vocabulary that it can communicate with other agents (e.g. request services, respond to requests, etc.), and negotiate with such agents (e.g. modify its request if the other agent cannot satisfy them). Last, but not least, the agent must be *personalizable*, i.e. it must come equipped with a very simple, but expressive language, that allows a novice computer user to express complex commands of the form "Monitor property p. If p changes in a specified way ch, then take action a." We are developing different types of abstractions that characterize these notions of awareness, collaboration, and personalizability and develop techniques to efficiently represent them and manipulate them computationally.

The agent desktop

The Agent Desktop is the most important part of **CATS** system. Each Agent Desktop will contain a variety of information about different types of agents, and may be personalized to the needs of one single user. For example, an Army commander, planning a route for a tank column, can have *his copy* of the Agent Desktop specifically configured for his needs. This configured version would then differ from the personalized versions of the Agent Desktop used by members of other echelons, staffs, forces, etc.

To achieve such functionality, the Agent Desktop manipulates three kinds of abstractions. As shown in Figure 1. each actual software agent will have consist of its underlying implementation (program code), plus instances of each of these three abstractions.

1. **Agent Property Abstraction (APA):** Each agent has certain intrinsic properties. These properties include:

    - the node address where the agent resides,

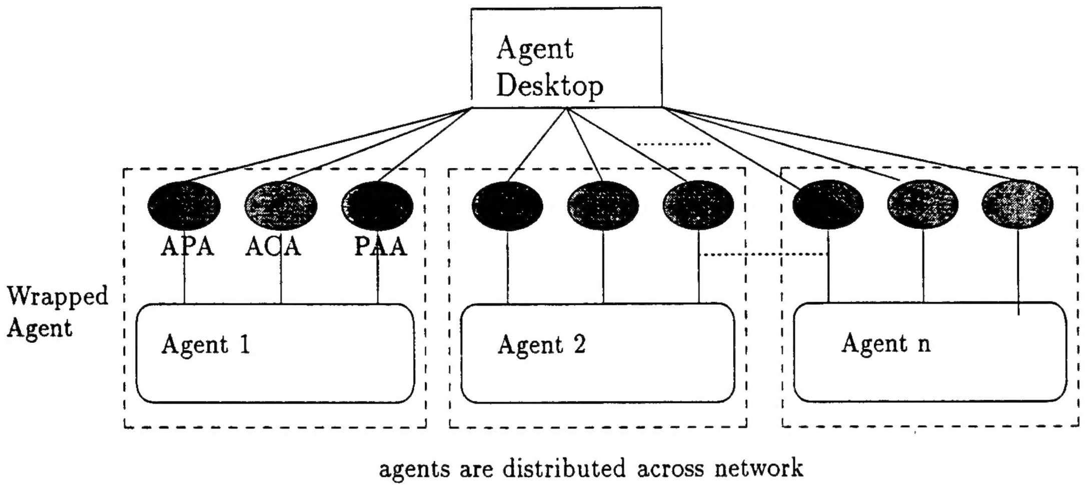

Figure 1: Architecture of Agent Desktop

- the method by which the agent can be invoked (e.g. by passing a certain control string to the server at the above location)
- the reliability of the agent.
- other locations where an identical agent resides (in case the preferred location is "down"),
- the format in which the agent expects input, and the format in which it returns output,
- the semantics of the agent's output (e.g. the units – centimeters vs. inches – in which different output values are returned),
- other agents that have the capability to process this agent's output.
- information on whether the agent is an *active* agent (i.e. one that is always "on" and that may activate other agents) as opposed to a *passive* agent (e.g. one that is usually "off").
- Information on whether the agent is mobile or not (i.e. can the agent "migrate" to other sites and execute commands, and if so, under what conditions is this feasible).

2. **Agent Collaboration Abstraction (ACA):** Each agent has certain properties that are relevant to collaboration. For example, we have already noted that agents will communicate with each other. However, the (human) user must to configure the way in which the agents "assimilated" within his desktop interact with others. If he chooses not to do so, we will define default parameters.

For this purpose, we have developed a simple rule-based framework that specifies the conditions under which one agent may invoke another one, together with the formal mechanism used in such an invocation. In its simplest form, a rule of this form may look like:

$$\texttt{can\_invoke(Agent1,Agent2,Intype,Outtype)} \leftarrow \texttt{Condition.}$$

Of course, such rules will be expressed using a graphical user interface (so that a user will not have to be burdened with the responsibility of writing rules). When the Army commanders Desktop includes such a rule, it means that `Agent1` may call `Agent2` with a string having the format of `Intype` and the output returned by `Agent2` will have the format specificed in `Outtype` . Thus, for example, a specification of the form

$\texttt{can\_invoke(RP1,RP2,}$ "[x/int, y/int] : 1 mile", "{[x/int, y/int] : 0.1 mile})"
$\leftarrow$ Condition.

says that when the `Condition` is satisfied, route planner `RP1` may invoke route planner `RP2`. The parameter `[x/integer, y/integer]:1 mile` may say that the inputs to `RP2` which are provided by `RP1` assume that each pixel represents a 1 mile by 1 mile region, while the route provided as output by `RP2` must generate a route at a finer granularity of 0.1 mile by 0.1 mile.

3. **Personalized Agent Abstraction (PAA):** It is entirely possible that there are certain tasks that a user performs often. For example, an Army commander planning a troop movement into Bosnia, may want to set up a *personalized agent* that does the following tasks every hour:

   (a) It invokes a *weather monitoring agent* to see if there are any major climactic changes expected along the route the the troop movements are planner.

   (b) It invokes a *satellite image intelligence* agent to determine if any substantial enemy troop movements have occurred in the region of operations, and if so, it checks to see if any of these opposing forces poses a threat to friendly forces by invoking a *Threat Assessment* agent.

   (c) If the threat posed above is substantial, it alerts one or more relevant officers (e.g. the commander, and selected staff) by emailing them or faxing them, an alert. This alert may well include also, a number of alternative courses of action (COAs) proposed by a Planning Agent.

   To set up such a personalized agent, the user needs to specify the logical sequences of actions described above. This is done by expressing the above interaction as *rules* of the form:

$$do(< action >, T) \leftarrow \texttt{Condition.}$$

This rule specifies that the action named `action` should be executed at all times `T` when the condition is satisfied. The content of the action itself is specified as a *Hierarchical Task Network* (HTN). The formal definition of such HTNs is beyond the realm of this short report – see [1] for a detailed discussion of the formalism of this kind of planning system.

Scalable-Backend Support for CATS

To support the needs of **CATS** and its users, the system must be implemented such that the agent architecture is supported computationally by a memory system capable of supporting the knowledge-based needs of intelligent agents, while still allowing the performance, security, distributed processing and scaling characteristics of current database systems. To date, knowledge-based systems have failed to allow these scaling and other capabilities, while agent systems based on databases have been very limited in their semantic and representational power.

CATS will require support for the reasoning needed by such an intelligent information system, will need to outperform current data-base systems, and will need the scaling and backend capabilities that will allow it to run on a wide variety of platforms – single processor, distributed network, and parallel supercomputers. To achieve this, we are focusing on providing scalable backend capabilities in a form usable by a wide variety of client agents using various emerging standards of agent protocols (KQML, GCCS, JTF Ref Architecture, etc.).

CATS is being supported using knowledge-based, rather than data-based, technology to provide the inferencing capabilities and semantic functions needed by intelligent agent systems. We have been developing both an implementation and underlying computational science of the development of very large knowledge bases. In particular, computationally, the networks represented in such knowledge bases are DAGs, with the links representing paths which are used to inherit information. The search on such an inheritance network can be very computationally intensive but we have proven them to be efficiently parallelizable. In our work, search engines have been implemented for AI knowledge bases [8, 9] and we have also shown that the parallelization techniques developed for these problems can be applied to searches on large information networks such as the world wide web, the military web, message repositories, and the like.

We are exploiting these techniques in a knowledge-representation backend system, known as Parka. Parka is implemented in C and has been run on a large variety of computing platforms ranging from small personal computers to the largest parallel supercomputers. For example, Parka supports case-based planning systems being developed for the Arpa Rome Laboratories Planning Initiative[4]. In experiments, we have shown that for a knowledge base containing over 2 million assertions , complex queries could be processed with times ranging from about about 200 milliseconds for a simple query (about 5 conjuncts and two restrictions) to about 4 seconds for one with 14 variables and 11 restrictions on a single processor Sparc 20. On an IBM SP/2 with 16 processors, times ranged from 29 milliseconds to about .5 seconds for the worst case.

We've also tested the system on the knowledge ontology of CYC[3] (about 50,000 frames in our systems) and the ISI Penman dictionary (about 80,000 frames)[7]. On those KBs we seem to be even faster, although we don't have a good enough test corpus to really test against.

The significant performance increases in the PARKA knowledge representation system, have made heavy use of database concepts in our implementations. We have recently perfected a mechanism by which we can actually implement our inheritance algorithms within a DB – that is, we modify the DB code to add our algorithms, and we build some extra tables that encode the KB relations. In addition, we can directly call remote databases (using SQL or other standard DB languages) for data. We can then do inheritance, recognition, and structure matching directly in the DB system. The two advantages of this is are (i) we no longer need to import a DB into a different format to add KB capabilities, and (ii) we get access control, secondary memory management, security, etc. "for free" (i.e. using the DB). Essentially, this means we are no longer limited to specialized machines, and that the size of the KBs we can manipulate are limited only by the secondary storage capabilities of the system

To support the needs of **CATS**, a number of extensions will be needed for the current Parka system. It must be made to work with a more complex (commercial quality) database implementation, a distributed processor version must be realized and tested, and query optimization techniques must be developed to allow the system to perform real-time query processing for the specialized needs of these distributed agent systems.

Conclusions

This report outlines the current status of a large ongoing effort to provide both a formal basis and a scalable platform for the development of large-scale intelligent agent applications. The goal of this report was to describe some of the areas, both theoretical and tool-based, which we feel are critical to the development of such agents. We believe these tools are now in place and ready for development, and this is the focus of our follow-on work now being supported by the Army Research Laboratory.

References

[1] K. Erol, J. Hendler, and D. Nau, Complexity Results for Hierarchical Task-Network

Planning, *Annals of Mathematics and Artificial Intelligence,*, 1997 (in press).

[2] A. Brink, S. Marcus and V.S. Subrahmanian. Heterogeneous Multimedia Reasoning. IEEE COMPUTER, 28, 9, pps 33–39, Sep. 1995.

[3] Lenat, D.B. and Guha, R.V., "Building Large Knowledge-Based Systems", Addison Wesley, Reading, Mass., 1990.

[4] J. Hendler, K. Stoffel and A. Mulvehill, High Performance Support for Case-Based Planning Applications, in A. Tate (ed) *Advanced Planning Technology,* MIT/AAAI Press, Menlo Park, CA., USA, May 1996

[5] J. Lu, A. Nerode and V.S. Subrahmanian. Hybrid Knowledge Bases, accepted for publication in: IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING.

[6] S. Marcus and V.S. Subrahmanian. Foundations of Multimedia Database Systems, JOURNAL OF THE ACM, Vol. 43, 3, pps 474–523, 1996.

[7] K. Knight and S. Luk, "Building a Large Knowledge Base for Machine Translation," AAAI Twelfth National Conference on Artificial Intelligence, 1994.

[8] K. Stoffel, J. Hendler and J. Saltz, High Performance Support for Very Large Knowledge Bases, *Proc. Frontiers of Massively Parallel Computing*, Feb. 1995.

[9] K. Stoffel, J. Hendler, J. Saltz and W. Andersen, *Parka on MIMD supercomputers*, in J. Geller (ed) *Parallel Processing in AI: II*, (in press).

[10] V.S. Subrahmanian. Amalgamating Knowledge Bases, ACM TRANSACTIONS ON DATABASE SYSTEMS, 19, 2, pp. 291–331, 1994.

2 Publications and Lectures

I. INVITED CONFERENCE ADDRESSES.

1. Invited Talk, 1997 Intl. Workshop on Logic Programming and Non-Monotonic Reasoning, Dagstuhl, Germany. *(V.S. Subrahmanian).*

2. Keynote Address: 1998 Intl. Workshop on Query Processing and Multimedia Issues in Distributed Systems, August 26-27, Vienna, Austria. *(V.S. Subrahmanian).*

3. Invited Lectures on Logic-Based Heterogeneous Information Integration, GULP Summer School in Logic Programming, Acquafredda di Maratea, Italy, Sep. 7-12. 1998. *(V.S. Subrahmanian).*

4. Invited Lectures on Intelligent Agents, GULP Summer School in Logic Programming, Acquafredda di Maratea, Italy, Sep. 7-12, 1998. *(S. Kraus).*
5. Invited Lecture on Heterogeneous Agent Systems, 1999 Intl. Symposium on Methodologies for Intelligent Systems (ISMIS-99), Warsaw, Poland, June 8-11, 1999. *(V.S. Subrahmanian).*
6. AI for the Internet, Invited Talk, Second International and Interdisciplinary Workshop on Intelligent Information Integration, Brighton, UK, Aug. 1998. *(J. Hendler).*
7. IDA Special briefings, Warfighter Information Systems, to Gen. Welch, Pres., IDA 12/97; To AF Chief Scientist Dan Hastings, IDA 2/98. *(J. Hendler).*
8. Developing Intelligent Agents - The future of AI planning systems, Invited Talk, 3 Simposio Basiliero de Automacao Inteligente (3rd Brazilian Symposium on Intelligent Automation), Vitoria, Brazil, Sept., 1997. Knowledge Representation for the World Wide Web, NASA Ames research Center, California, Feb. 1998. *(J. Hendler).*
9. Invited lecture on Strategic Negotiation and Cooperation Among Autonomous Agents, Sixth Scandinavian Conference on Artificial Intelligence, Helsinki, August, 1997 *(S. Kraus).*
10. Invited lecture on IMPACT: The Interactive Maryland Platform for Agents Collaborating Together, Israeli seminar on Artificial Intelligence, March 27, 1998, Israel. *(S. Kraus).*

II. JOURNAL PAPERS/SUBMISSIONS.

1. K. Arisha, S. Kraus, F. Ozcan, R. Ross and V.S.Subrahmanian. *IMPACT: The Interactive Maryland Platform for Agents Collaborating Together.* Submitted for publication, Nov. 1997.
2. Dix. Jürgen and V.S.Subrahmanian, *Meta_Agent Programs*, Submitted for publication.
3. S. Luke and J. Hendler, Web Agents that Work. IEEE Multimedia , 4(3), 1997.
4. J. Hendler. Intelligent Agents — Where AI meets Information Technology, (i) IEEE Expert , December, 1996. ii. Reprinted in Proceedings, 3rd Brazilian Symposium on Intelligent Automation, Victoria, Brazil, Sept, 1997.
5. T. Eiter, V.S. Subrahmanian and G. Pick. Heterogeneous Active Agents. Submitted to *Artificial Intelligence* journal, March 1998.

6. A. Dekhtyar and V.S. Subrahmanian. Hybrid Probabilistic Programs. Submitted to *Journal of Logic Programming*, Jan. 1998. Under revision as per referee comments, July 1998.

7. S. Kraus, Y. Sagiv and V.S. Subrahmanian. Representing and Integrating Multiple Calendars. Submitted for journal publication.

8. V.S. Subrahmanian. Nonmonotonic Logic Programming, accepted for publication in IEEE Transactions on Knowledge and Data Engineering.

9. A. Brogi, V.S.Subrahmanian and C. Zaniolo. Modeling Sequential and Parallel Plans: A Deductive Database Approach, submitted for journal publication. O. Shehory and S. Kraus. Methods for Task Allocation via Agent Coalition Formation. Artificial Intelligence journal, 101(1-2):165-200, 1998.

10. S. Kraus, K. Sycara and A. Evenchik. Reaching agreements through argumentation: a logical model and implementation, Artificial Intelligence journal, (to appear.)

11. O. Shehory and S. Kraus. Feasible Formation of Stable Coalitions among Autonomous Agents in Non-super-additive Environments, Computational Intelligence, (to appear.)

12. O. Shehory, S. Kraus and O. Yadgar. Goal Satisfaction in large scale agent-systems: A transportation example. in A. Rao, M. Singh and M. Wooldridge editors, Intelligent Agents V (to appear). (Won the ATAL98 best paper award)

13. K.S. Candan, B. Prabhakaran and V.S. Subrahmanian. Collaborative Multimedia Documents: Authoring and Presentation accepted for publication in *Intl. J. of Intelligent Systems.*

14. K.S. Candan, B. Prabhakaran and V.S. Subrahmanian. Retrieval Schedules Based on Resource Availability and Flexible Presentation Specifications. ACM Multimedia Systems Journal, Vol. 6, Nr. 4, pps 232–250.

15. K.S. Candan, E. Hwang and V.S. Subrahmanian. An Event-Based Model for Continuous Media Data on Heterogeneous Disk Servers. ACM Multimedia Systems Journal, Vol. 6, Nr. 4, pps 251–270.

16. L. Golubchik, S. Marcus and V.S. Subrahmanian. Sync Classes: A Framework for Optimal Scheduling of Requests in Multimedia Storage Servers, Accepted for publication in *IEEE Transactions on Knowledge and Data Engineering.*

17. O. Seeliger and J. Hendler, Supervenient Hierarchies of Behaviors in Robotics. Journal of Experimental and Theoretical AI , 9(2/3), 1997

III. CONFERENCE PAPERS/SUBMISSIONS.

1. K. Stoffel, M. Taylor and J. Hendler, "Efficient Management of Very Large Ontologies," Proc. AAAI-97, Providence, RI, 1997.
2. L. Nunes de Barros, R. Benjamins, and J. Hendler, "Par-kap: A knowledge Acquisition Tool for Building Practical Planning Systems," Proc. IJCAI-97, Nagoya, Japan, 1997. ii. reprinted in Proc. The Ninth Dutch conference on Artificial Intelligence (NAIC-97) K. van Marcke, W. Daelemans (eds), University of Antwerp, Belgium, November, 1997.
3. S. Luke, L. Spector, J. Hendler and D. Rager, "Ontology-Based Web Agents", Proc. Agents 1997, San Mateo, CA. 1997.
4. Sean Luke, Charles Hohn, Jonathan Farris, Gary Jackson, and James Hendler, Co-evolving Soccer Softbot Team Coordination with Genetic Programming, Proceedings of the First International Workshop on RoboCup, Nagoya, Japan, 1997.
5. V. Manikonda, P. S. Krishnaprasad, and J. Hendler, "Languages, Behaviors, Hybrid Architectures and Motion Control" (1997)," Essays in Mathematical Control Theory (in honor of the 60th birthday of Roger Brockett), (eds. John Baillieul and Jan C. Willems), Springer-Verlag.
6. D. Rager, J. Hendler and A. Mulvehill, ForMAT and Parka: A technology integration experiment and beyond," Proc. Intl Conference on Case-Based Reasoning ,Providence, RI, 1997.
7. B. Kettler , J. Hendler and K. Sanders, "The Case for Graph-Structured Representations," Proc. Intl Conference on Case-Based Reasoning , Providence, RI, 1997.
8. S. Luke, L. Spector, J. Hendler and D. Rager, "Ontology-Based Web Agents", Proc. Agents 1997, San Mateo, CA. 1997.
9. Sean Luke and Lee Spector, A Comparison of Crossover and Mutation in Genetic Programming, Proc. Genetic Programming (GP97), Stanford, 1997.
10. P. Emmerman, J. Hendler and V.S. Subrahmanian. CATS: An architecture for scalable intelligent agent applications, *Proc. 1997 Bar-Ilan Symposium on Foundations of Artificial Intelligence*, Israel, June 1997.
11. P. Bonatti, S. Kraus, J. Salinas, V.S. Subrahmanian. Data-Security in Heterogeneous Agent Systems, in M. Klusch editor, Cooperative Information Agents, Springer-Verlag, Springer-Verlag, 1998, pp. 290–305.

12. K. S. Candan, E. Lemar, V.S. Subrahmanian. *Management and Rendering of Multimedia Views*, Proc. 1998 Intl. Workshop on Multimedia Information Systems, Sep. 1998 (to appear), Springer Verlag Lecture Notes in Computer Science.

13. J. Schafer, T. J. Rogers, J.A. Marin. Networked Visualization of Heterogeneous US Army War Reserve Readiness Data, Proc. 1998 Intl. Workshop on Multimedia Information Systems, Sep. 1998 (to appear), Springer Verlag Lecture Notes in Computer Science.

14. E. Bertino, S. Jajodia, P. Samarati and V.S. Subrahmanian. (1997) *A Unified Framework for Enforcing Multiple Access Control Policies*, *Proc. 1997 ACM SIGMOD Conf. on Management of Data*, Tucson, Arizona, May 1997.

15. S. Jajodia, P. Samarati and V.S. Subrahmanian. (1997) *A Logical Language for Expressing Authorizations*, *Proc. 1997 Oakland Conf. Computer Society.*

16. A. Dekhtyar and V.S. Subrahmanian. (1997) *Hybrid Probabilistic Programs*, *Proc. 1997 Intl. Conf. on Logic Programming,* Leuven, Belgium, July 8-12, 1997/

17. T. Eiter, J. Lu and V.S. Subrahmanian. (1997). *Computing Non-Ground representations of Stable Models*, *Proc. 1997 Intl. Conference on Logic Programming and Non-monotonic Reasoning*, Dagstuhl, Germany, July 1997. Springer Verlag.

18. E. Hwang, P. Prabhakaran, and V.S. Subrahmanian. *Distributed Video Presentations*, in Proc. 1998 IEEE Intl. Conf. on Data Engineering.

19. S. Adali, P. Bonatti, M.L. Sapino and V.S. Subrahmanian. *A Multi-Similarity Algebra*, Proc. 1998 ACM SIGMOD Conference on Management of Data, June 1998, Seattle, WA.